51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年08月
总第017期



架构师成长历程：行到水穷处，坐看云起时

程序员成熟的标志

8月编程语言排行榜：C语言助力火星登陆

developer.51cto.com

目录

2012.08 Contents



编程排行 Billboard

3 8月编程语言排行榜：C语言助火星登陆

专题报道 《架构师成长历程》

7 架构师成长:行到水穷处,坐看云起时

9 陈皓：带队、沟通、成长与变化

11 淘宝专家嗷嗷：前端开发应如何去做

13 一淘清无：Nginx_lua的测试及选择

16 合格的架构师是如何炼成的

17 架构师的成长历程：实践出真知

技术热点 Techlogy hot

18 程序员成熟的标志

20 身居中国的法国程序员谈Python/PHP

22 大数据时代 数据分析师的尴尬

23 数据分析师的级别

24 代码整洁之道

26 你的代码值多少钱？你算过吗

28 国外程序员是如何准备面试的

30 全世界有多少Java开发者?

32 开发人员眼中最好的代码编辑器

34 做有中国特色的程序员

51CTO.com
技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡 张伟 林师授 封面设计：钱靓

联系方法：pengfan@51cto.com 010-68476606（分机 8058）

出版日期：2012 年 8 月 20 日

欢迎广大读者投稿

8月编程语言排行:C语言助火星登陆

2012 年 8 月，Tiobe 发布了 2012 年 8 月最新一期编程语言排行榜。C 语言继续其上升的趋势，Java 份额也在进一步下降。而 C#、Python、Perl 等语言保持原来的位置。今天我们要谈到的是 C 语言，正是他帮助“好奇号” 2012 年 8 月 6 日登陆火星。

2012 年 8 月，Tiobe 发布了 2012 年 8 月最新一期编程语言排行榜。C 语言继续其上升的趋势，Java 份额也在进一步下降。而 C#、Python、Perl 等语言保持原来的位置。今天我们要谈到的是 C 语言，正是他帮助“好奇号” 2012 年 8 月 6 日登陆火星。以下是前 20 名编程语言排行：

Position Aug 2012	Position Aug 2011	Delta in Position	Programming Language	Ratings Aug 2012	Delta Aug 2011	Status
1	2	↑	C	18.937%	+1.55%	A
2	1	↓	Java	16.352%	-3.06%	A
3	6	↑↑↑	Objective-C	9.540%	+4.05%	A
4	3	↓	C++	9.333%	+0.90%	A
5	5	=	C#	6.590%	+0.55%	A
6	4	↓↓	PHP	5.524%	-0.61%	A
7	7	=	(Visual) Basic	5.334%	+0.32%	A
8	8	=	Python	3.876%	+0.46%	A
9	9	=	Perl	2.273%	-0.04%	A
10	12	↑↑	Ruby	1.691%	+0.36%	A
11	10	↓	JavaScript	1.365%	-0.19%	A
12	13	↑	Delphi/Object Pascal	1.012%	-0.06%	A
13	14	↑	Lisp	0.975%	+0.07%	A
14	26	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.877%	+0.41%	A
15	15	=	Transact-SQL	0.849%	+0.03%	A
16	18	↑↑	Pascal	0.793%	+0.13%	A
17	11	↓↓↓↓↓↓	Lua	0.726%	-0.64%	A-
18	16	↓↓	Ada	0.649%	-0.05%	B
19	22	↑↑↑	PL/SQL	0.610%	+0.08%	B
20	29	↑↑↑↑↑↑↑↑↑	MATLAB	0.533%	+0.09%	B

美国东部时间 2012 年 8 月 6 日凌晨 1 时 30 分(北京时间 6 日 13 时 30 分)，新型火星探测器“好奇”号计划着陆火星表面。作为迄今为止设计最为复杂精密的火星探测器，好奇号探测车采用的是风河公司业界领先的 VxWorks 实时操作系统(RTOS)。好奇号采用 VxWorks 系统完成被称为 EDL(进入火星大气层、下降以及着陆)的复杂着陆过程。由于宇宙飞船安全着陆需要绝对的精确度，这一过程被称为“恐怖七分钟”。从 2011 年 11 月 26 日火箭离开地球那一刻起一直到任务完成，VxWorks 作为火星探测车的核心操作系统，将在本次具有历史意义的活动上发挥至关重要的作用。整个火星车里有 50 万行程序代码，运行在 VxWorks 操作系统上，精准无误，以毫微妙计算。

VxWorks 与 C 语言

VxWorks 是美国 Wind River System 公司(以下简称风河 公司 ，即 WRS 公司)推出的一个实时操作系统。Tornado 是 WRS 公司推出的一套实时操作系统开发环境，类似 Microsoft Visual C，但是提供了更丰富的调试、仿真环境和工具。

Wind River 和 VxWorks 相信很多嵌入式程序员都不会陌生。VxWorks 支持 C 语言或者 C++ 进行开发，并配备了叫 Tornado 的集成开发环境。Tornado 缺省使用 GNU 编译器，支持 Ansiiso C, C++，以及 AT&T 语法汇编(非 Intel)。由于 C++ 很庞大，而且在不断发展，所以有些标准 C++ 的特征可能 GNU 不支持(基本没有编译器支持所有 C++ 特征)。程序员一般只要掌握常用部分就能编出不赖的程序了。inb(),outb() 是 turbor cdos 下的，不属于标准 C。Tornado 在 bsp 中提供了相应的 sysinbyte(),sysoutbyte() 等函数，参考 bsp 下的 sysalib.s。

Tornado 结构的专门设计为开发人员和第三

2012 年 8 月编程语言排行榜：C 语言助力火星登陆 II

方工具厂商提供了一个开放环境。已有部分应用程序接口可以利用并附带参考书目，内容从开发环境接口到连接实现。

Tornado 包括强大的开发和调试工具，尤其适用于面对大量问题的嵌入式开发人员。这些工具包括 C 和 C++ 远程源级调试器，目标和工具管理，系统目标跟踪，内存使用分析和自动配置。另外，所有工具能很方便地同时运行，很容易增加和交互式开发。关于 VxWorks 组成部分，请参考《VxWorks 组成部分》一文。

新一代开发工程助推器——Wind River Workbench

集成化的开发环境意味着一个开发工作团队运用同一种语言和工具，不论在任何开发项目的任何工作阶段，都可以共同为所有的开发项目做出贡献。

Wind River Workbench 提供了一种开放式、基于标准的集成化设备软件开发环境。它使得企业能够标准化它的整个设备软件开发过程到一个公共平台下，以其独特的将可靠性，功能性以及端对端集成性整合到一起方式，大幅优化了工程人员、项目团队和企业的工作效率。

与 VxWorks 6.x 系列配套的 IDE 组件是 Wind River Workbench，这也是 Wind River GPP VxWorks 版的重要组成部分之一。

比如 Wind River Workbench 2.2 可以最大限度地优化电子产品制造企业的人力资源并使他们做出最佳的表现。其中的诀窍在于把他们的时间、能力和经费都用于增加电子产品的特殊使用价值，而不是把这些资源浪费在低级、普通、一般的工作中。凭借在设备软件开发领域多年积累的丰富经验，风河平台之中包括了功能齐全、相互兼容并且运行十分可靠的中间件，可以满足许多不同行业标准的特殊需求。而且，由于多年来许多行业都在广泛应用 VxWorks 实时操作系统，已经形成了一个稳定而良性的合作伙伴群体，这就使得 Wind River GPP VxWorks 版具有非常好的完备性，很快就可以得到开发项目所需的所有资源，从而可以更好地控制用户的经费预算和产品上市时间。可以说，有了 Wind River GPP VxWorks 版，再加上 Wind River 的合作伙伴全体和专业的培训以及针对用户开发项目的指导方案，你就可以解决设备软件开发的所有问题了。

登陆火星用了多少代码？

根据 51CTO 记者了解到的情况，本次 Curiosity 好奇号登陆火星，其包含 250 万行 C 语言代码。这些代码运行在 BAE 制造的 RAD750 处理器上，他们包括 150 个独立模型，每个承担不同的功能。高度耦合的模块被抽象成组件，被置顶一个特定的功能或者行为。这些组件被进一步组合为层，整个火星登陆车包括不超过 10 个顶级层组织。

好奇号项目中的软件开发经验

浩大的好奇号项目其实美国创意产业和商业力量的结合，并且 NASA 的项目管理者自有一套创新和商业逻辑。在长达 9 年，高达 25 亿美元的投资中，火星科学实验室项目（好奇号火星登陆项目）其实是美国创意产业和商业力量的结合。好奇号的登陆模拟团队同迪斯尼动画公司保持密切联系，从而使得 3D 登陆模拟更加精确。其它的一些系统解决方案有来自石油公司的，深海钻探公司的，甚至是农业公司的技术。

2012 年 8 月编程语言排行榜：C 语言助力火星登陆 III

这一浩大的工程是如何完成的？ NASA 的管理者自有其一套创新和商业逻辑。

允许在研发过程中的奇思妙想

一个典型的项目想法，可能是起源于一张餐巾纸上的草图，然后变成 3D 模拟模型，然后再变成生产模型。

这一项目进行过程中，研发人员充满了敏锐的直觉和想象力。有时候，他们会去想象在太空中移动登录器的各个部件，构建想法。

没有过于疯狂的想法，真的！

在项目的最开始阶段，没有批评想法的习惯。研发人员，凭空想出任何点子，而其他人员不是去评价这个点子有多么的疯狂甚至是批评，而是去思考如何去改进这个点子。

乐于从草案开始

就像好的想法经常来源于之前的错误和过去的经验一样，项目的研发人员更倾向于从过去的草案中起步，而不是从头开始。

不要陷于复杂的事物，将其分解为可控的最小单元

因为有太多的复杂的、不同的系统在好奇号的平台上。项目的工程师们都是在相互区别的区分的系统中完成他们的工作。他们将会把项目分成无数的子系统，子部件，一遍解决单独的子问题。

保持开门策略

无论是管理层，还是基层员工，每个人之间都能保持一种坦诚的沟通，并且能就具体问题进行具体的分析。

不要重复发明轮子

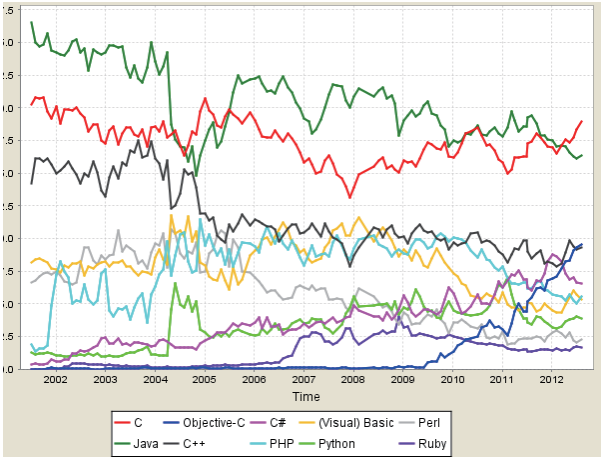
火星登陆项目从现实商业项目中学到很多经验，当研发人员发现某一部分的问题，已经有现成的解决方案的时候，他们要做的不是重复发明轮子，而是直接拿来用。

允许没有伤害的小理由

在好奇号项目的软件中，以为工程师曾经在软件代码中添加了一段几年其过世的父亲的代码。这些没有伤害的一些事情，被允许从而使得研发者对该项目充满了感情。

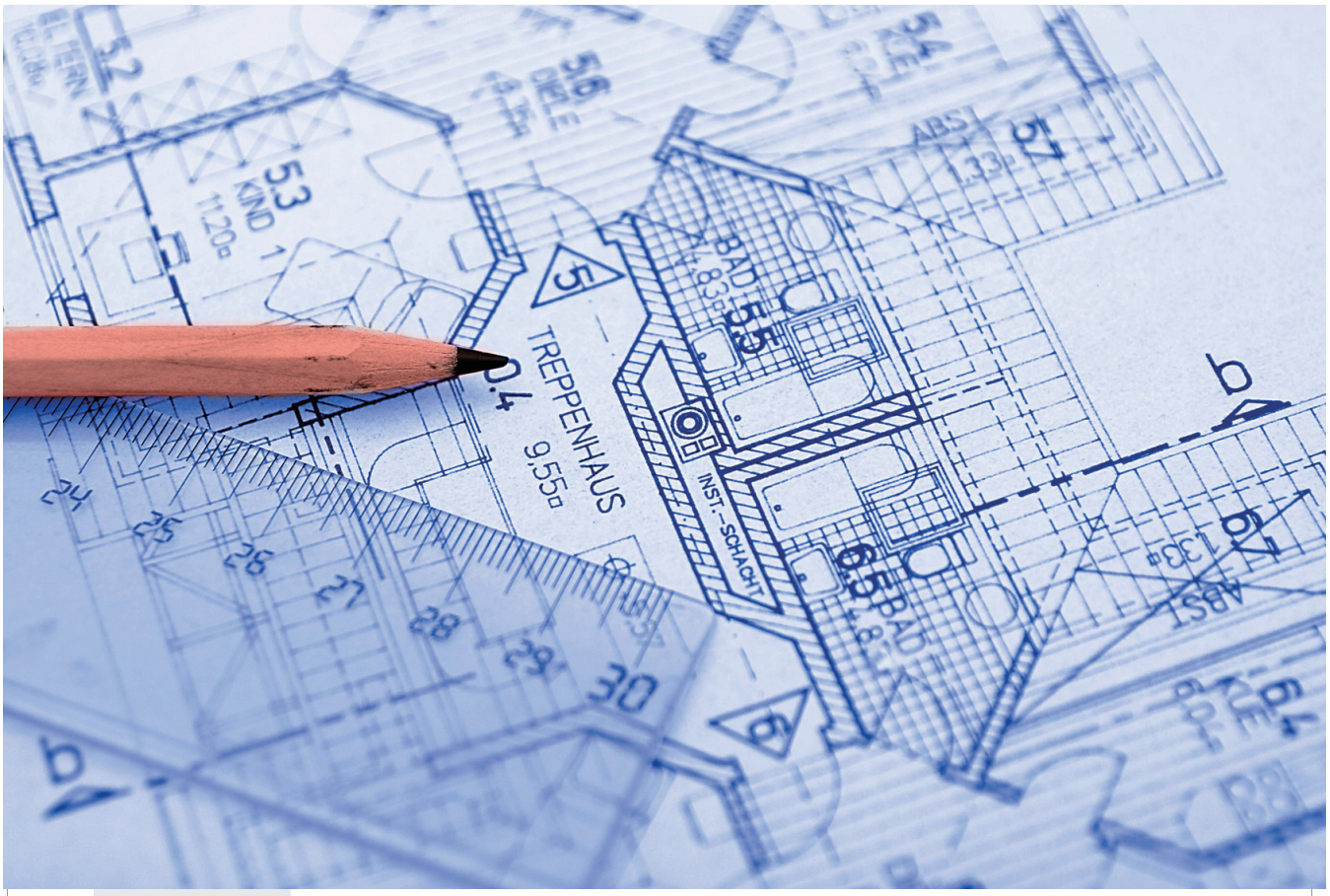
下面是本期编程语言排行榜的其他排名数据和趋势走向。

前 10 名编程语言走势图



下面是第 50 到 100 的编程语言排名

(Visual) FoxPro, Apex, bc, BETA, Boo, CFML, CHILL, CL (OS/400), Clean, Clojure, Curl, Dart, Dylan, Eiffel, Emacs Lisp, F#, Gambas, Go, Groovy, Icon, Inform, Informix-4GL, J, JavaFX Script, LabVIEW, LabWindows/CVI, Ladder Logic, Maple, Modula-2, MUMPS, NATURAL, Oberon, OCaml, Occam, OpenCL, Oz, PowerShell, Processing, REALbasic, REXX, S, S-PLUS, SPARK, SPSS, Tcl, VBScript, Verilog, VHDL, X10, xBase ■



对于任何一个程序员来说，架构师都是一个令人向往的角色。可以说除了少部分对程序开发有着极度的热爱并愿意为之奋斗终身的程序员，而大部分的程序员都只是把写代码作为一个获得职业提升的一个必要的累积阶段。

在不断的开发和学习各种知识，经验，在自我培养和技能强化中不断的提升，为他们的未来中成为一名合格的架构师打下牢固的基础。

一名优秀的架构师，他们都是带着极度的热情，踏上正确的路上，不断的坚持！这就是一名优秀架构师的态度。在 51CTO 本期专题中，有小编有幸采访到了 51CTO 云计算架构师峰会的邀约讲师，前任金蝶总体架构部 SOA 架构师，现淘宝网高级专家彭渊（花名：千峰）老师，跟我们一起聊聊关于他的架构师历程。

51CTO 开发频道语

编者按

这次来讲 MVC 中最后的 M。Model，几乎可以说是网页应用的核心。之前课程提到过网页应用是由数据库驱动，而在很多场景，数据库 = M ； M = 数据库。

架构师成长:行到水穷处,坐看云起时

【51CTO 专稿】对于任何一个程序员来说，架构师都是一个令人向往的角色。可以说除了少部分对程序开发有着极度的热爱并愿意为之奋斗终身的程序员，而大部分的程序员都只是把写代码作为一个获得职业提升的一个必要的累积阶段。在不断的开发和学习各种知识，经验，在自我培养和技能强化中不断的提升，为他们的未来中成为一名合格的架构师打下牢固的基础。

一名优秀的架构师，他们都是带着极度的热情，踏上正确的路上，不断的坚持！这就是一名优秀架构师的态度。下面小编有幸采访到了 51CTO 云计算架构师峰会的邀约讲师，前任金蝶总体架构部 SOA 架构师，现淘宝网高级专家彭渊（花名：千峰）老师，跟我们一起聊聊关于他的架构师历程。



前任金蝶总体架构部 SOA 架构师，现淘宝网高级专家 千峰

千峰简介：从业 Java 技术领域十多年，撰写过多款开源软件，拥有软件著作权的代表作有：BS 系列软件（包括财务进销存、OA 产品、CRM 等）、FMS 视频会议、Flash 网站生成软件。其中 BS 系列软件获得过风险投资， FMS 视频会议获得金山举办的首届中国优秀软件创新大赛优秀奖。其中《BS 财务进销存》《FLASH 网站生成软件》在华军、天空可免费下载。《淘宝分布式并行计算四合一框架 Fourinone》是云计算方面作品，目前已经通过压测并用于淘宝业务场景，wordcount 和排序等经典指标均优于 hadoop。

详见技术博客（可下载开发包和 demo）：http://3503265.blog.51cto.com/

专长与研究领域：千峰踏入 IT 行业至今差不多 12 年了，一直做 Java 技术领域，几乎 Java SE，Java EE，Java ME 都涉及过。除此外，他还比较熟悉 FLASH 音视频技术和少量的 C 技术，主要是因为 Java 在音视频和操作系统层面控制比较弱的原因。而在应用方面，B/S 结构的企业软件架构更是他的专长，并拥有一系列软件著作权。在淘宝先后在交易中心、中间件、核心系统部门，主要研究非法软件防范（秒杀器，炒信用，代拍，爬虫等）、大数据处理、语义分析、分布式计算、机器资源利用等领域。

（以下内容均为第一人称）

从对 IT 热爱到程序员生涯

我在高中时买了台中英文小霸王游戏机（那

架构师成长：行到水穷处，坐看云起时 ||

时有电脑的家庭很少），可以接在电视上玩，里面有个 Basic 编程的，我第一次学会了 goto 语句。大学时喜欢打电脑游戏，后来又觉的游戏打的好不算什么，当时流行做个人主页，有个漂亮的个人主页很有面子，于是我就买了本 FrontPage 的书，暑假关着门学做网页，穿插一些 js 动态效果和 applet 动画，看见流行 flash 动画，我又学习 flash 脚本。后来去找实习工作，别人说我搞的这些没用，他们不需要网页制作，他们要用 PB+SQL Server，然后我又继续学习数据库应用，一直到毕业，我自学了很多 VC++/MFC/ODBC 等东西，也学习了 Java。我大部分的编程都是靠自学，计算机课程和老师教的帮助比较小．我的第一份工作是基于 Java 编程的，还专门到 bea 公司进行了 J2EE 培训，后来就一直做了下来。



有朋友问过我，为什么只做 Java 编程，不去做 C 编程。实际上，如果仅仅是编程语法语句，任何一种语言我都可以很快学会。关键 Java 技术已经是一个从无线到桌面到 Web 到 ic 卡的完整的技术体系和规范，并生长出各种技术生态系统，如果要考虑快速落地实现一个系统，很多处理经验是要积累起来的。

从程序员到架构师的经历

“从感兴趣、学习、怀抱当比尔盖茨理想”到“走入这个行业靠编程谋生并成长为一个架构师”是一个曲折蜕变的过程，我记的好像是在 2004 年左右，架构师这个岗位从国外传入，很多公司出现了招聘架构师，以前比较多出现系统分析员。什么是架构师，各公司的标准也不一样，架构师更多是个岗位，你只要聘上了就是架构师。我自己的经历大概是这样的：

1）架构师要能 hold 的住一个软件或者系统的设计和开发

2002 年我到广东发展没有经验，被两个老板骗到南海软件园包吃住干过一段时间，当时去投标一个南海民政的项目，公司是打着北方一个分公司名义的皮包公司，找了几个毕业不久的学生当工程师，又找了一个头衔多的海龟说是他们的技术总监，凑成一只廉价成本的团队去忽悠上百万的项目，老板说这个项目副市长亲口告诉他已经交给我们做了，只需要招标走走形式。

由于公司主要靠两个老板的关系，没有自己的技术产品，当时老板从其他公司找来一套民政系统软件，Web 化的，老板说是其他公司一个资深点小伙子写的，说其实 1-2 个人就可以写了，没什么难的，叫我改成公司的产品去投标。当时接到这套完整的 Web 系统软件时，虽然是 Java 的，但是没有用到多少 j2ee 里面花哨的东西，都是 jsp+Javabean 这种简化的技术，也有控制器等角色，都是 jsp 实现的，里面复杂的页面流和大量 Javascript 脚本显示出开发者娴熟的 Web 编程功底，数据表可以自由搭建，里面有些简化的生成管理模块的工具。当时有种 hold 不住的感觉，感觉自己 Web 程序还写的太少，还未积累大量 Web 编程的……■本文未完，详细部分请参考原文

http://developer.51cto.com/art/201208/352900.htm

陈皓：带队、沟通、成长与变化

看过酷壳这个网站的朋友们应该对我们今天采访的故事主角不陌生，他就是酷壳的董事会主席兼 CEO 兼 COO 兼前端工程师兼开发人员兼美工兼主编的陈皓（@ 左耳朵耗子）。身为亚马逊中国的研发经理，他目前的工作是什么样的？他的成长过程中都有怎样的故事？

【51CTO 专访】刚刚在深圳结束的 ArchSummit 大会上，近千名工程师们以“架构师”之名聚集在一起，讨论着有关架构师的种种。程序员的世界就好像一个武林，人们来自不同的师承，不同的派别，而无师自通、跨门派的修行者也不在少数。而当你去一一接触他们，了解他们的时候，会发现每个人都有自己独特的故事。

看过酷壳这个网站的朋友们应该对我们今天故事的主角不陌生。他在酷壳上这样介绍自己：

“我不是高手，我也不是牛人，我只是在跟随着我的兴趣和喜好去学习一些技术。”



陈皓（@ 左耳朵耗子），14 年软件开发相关工作经验，8 年以上项目和团队管理经验，6 年的软件行业咨询经验。擅长底层技术架构，团队建设，软件工程，软件研发咨询，以及全球软件团队协作管理。对高性能，高可用性，分布式，高并发，以及大规模数据处理系统有一些经验和心得。喜欢关注底层技术平台和互联网行业应用。技术擅长 C/C++ / Java 和 Unix / Linux / Windows。现于 Amazon 中国任研发经理，负责电子商务全球化业务（全球开店）的研发。

上面的简介是他自己写的。当然，陈皓不会在个人简介中提到自己每天晚上从 10 点阅读到 12 点，再更新酷壳到凌晨 2 点这种事情，而这些细节，对于学习中的人们而言其实很关键。

51CTO 开发频道最近在筹备一个有关架构师成长的专题，一方面这是很多对自己职业成长方向不确定的开发者们都很关注的话题，另一方面也是为了我们在 10 月底的云计算架构师峰会做铺垫（陈皓也是我们会议上的分享嘉宾之一）。因此，我们会去接触很多架构师，去了解他们的经验，以及对架构师的看法。

不过，我觉得更重要的是去了解一种思路，一种观察、学习的方法，而这些都隐藏在他们的故事当中——

那些有关成长的故事。

以下是 51CTO 本次对陈皓的采访内容实录。

51CTO：您的个人简介我想就不用说了，先谈谈您现在的职责吧。

陈皓：有带队、沟通、成长与变化 II

陈皓：要说介绍的话，就说我是酷壳的董事会主席、CEO、COO、前端工程师、开发人员、美工、主编这些(笑)。

我现在在亚马逊，这边主要的职责，第一个就是带着团队做软件开发项目，另外一个是制定团队的计划，来年的计划。然后还负责一些招聘工作。基本上就这几大块。

51CTO：这边团队主要是一个支持类的，还是一个产品类的？

陈皓：就是一个研发，带着大家去做技术选型、技术架构设计，等等这些东西。

很多技术经理更多的是做计划、管人，但是我不一样，我喜欢技术，所以我更偏的做技术一些。

我在亚马逊做第三方商户全球开店的项目，目前在做的业务是 demand forecasting，亚马逊内部的库存需求预测，就是下个月我要卖多少东西，做机器学习。团队里面全都是 SDE（软件开发工程师）。也不分什么工。我们都是每个人负责一个功能，然后从头做到尾。从需求分析，设计，编码，测试，到运维，全部做。

51CTO：那最初产品定义是您来制定？

陈皓：产品定义我们有一个product manager，或是 Technical Project Manager，用户的需求会到他们这里汇总，然后他们会和研发团队讨论需求应该怎么实现，优先级是什么，怎么简化需求，等等。对于库存预测系统，公司内的采购经理就是我们的客户，这个东西怎么操作会比较好，预测的准不准，他可以进行哪些调整，等等。对于第三方商户的全球开店业务，由 Product Manager 去调查第三方商户的需求和现有的运营情况，然后给研发团队提需求。

有时候技术团队也要去见客户，但主要是内部的客户，去问他一些东西，怎么做，做成这样行不行。

51CTO：团队人员搭配是怎样的？

陈皓：团队里面基本上是这样：有一个人非常懂底层，C/C++ 出身的，对操作系统的底层非常熟，喜欢看代码专研底层；还有一个人是 J2EE、Java 出身，对 Java 的架构和各种框架如 Spring，Struts 这些都很熟；还有一个人对面向对象和软件设计这些比较熟悉。还有人对软件的前端设计比较熟，做过 Web Portal 的设计；还有人对机器学习的算法和方法非常熟；还有一个人也倾向于底层，但是稍微倾向于网络这边。这些人，每个人都有一块自己非常强的东西，所以他们合作起来也会很舒服，因为他们可以从别人身上学东西，而且自己也能领着别人去做点东西。每个人都有自己的领导力，每个人都有自己的成长空间。我觉得这是让团队比较和谐的原因。

另外，我只是一个支持性的角色，团队主导一切，我只是在旁边支持他们。

51CTO：他们愿意去学习自己不了解的领域。

陈皓：对。没有英雄。也没有闲人。每个人都有自己的长处，在这个长处上他可以领着别人做事情，同时也能从别人身上学到自己比较薄弱的东西。■

本文未完，详细部分请参考原文

http://developer.51cto.com/art/201208/353256.htm

编者按

在 2012 年 7 月 7 日阿里技术嘉年华上记者遇到阿里巴巴前端技术专家刘杰(花名：嗷嗷)。嗷嗷从事前端开发多年，期间做过设计转过产品，混过后端，写过《Web 标准设计》一书。

淘宝专家嗷嗷：前端开发应如何去做

目前就职于淘宝网，主要负责前端性能监控、优化等前端开发相关的工作。在会上嗷嗷以“如何发现前端的性能问题”为话题分享了如何通过日志来分析问题，如何通过 har 来分析问题以及通过监控来发现问题等内容。



很多人认为学习前端开发其实高中生甚至初中生都可以学，学不了 C++ 以及 Java 才来学前端开发，在他们的眼中前端开发就是网站制作的初级技能，没有什么技术含量，随便拿个 Dreamweaver 拖拖拉拉就是一个网页出来。

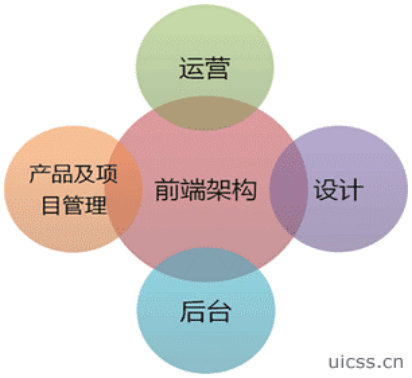
这不是前端开发！

严格来说，上面的观点也只能是一个网页设计的一个美工内容，根本就不会涉及性能，框架，脚本等待问题，所以也谈不上是前端开发者。那么前端开发到目前为止有没有一个明确的定义呢？有的！下面给出一个前端开发工程师需要掌握的技能表。

前端开发工程师不仅仅需要掌握 HTML+CSS+JS，还需要长时间的去了解，积累，测试各种主流的浏览器兼容问题，HACK 技术，熟悉各种框架，做 SEO 等等。

而前端开发的性能优化往往是后台的专属，但是作为前端开发的性能是极其的重要的。一个优秀的前端开发工程师写出来的代码能够让服务器压力小，承载能力翻倍，请求次数更少，合理的布局等等，这是一个普通的前端开发人员所做不到的。

我们从嗷嗷的介绍中就清楚，一个优秀的前端开发者，除了掌握前端开发的相关技能，对产品，后台，设计以及项目管理都需要了解的。一专多长，这才是前端开发人员的最终需求。下面给出一张图片，详细的解释了前端开发如何去一转多长。



做前端，先做“产品”！

对于一些交互设计，内容布局，良好的客户体验，往往关系着网站的成败。所以，一个好的页面，要从如何去做好这个产品开始，从做产品上掌握并提高客户的体验。比如我们在 51CTO 的云计算架构师峰会的购票页面里面填写了一个错误的邮箱，网站就能即时反馈输入错误的信息。如图：

淘宝专家嗷嗷：前端开发应如何去做 II



当然还有一些弹出提示层和 cookie 的记录等等，只要把客户的需求做好了，那么你这个“产品”就是成功的！

淘宝前端专家嗷嗷也是从设计，产品以及后台开始，他认为，从事前端开发在不同的阶段可能你需要了解到一些知识也不一样，比如你想深入了解后端的一些东西，那你如果作为一名前端的，所以你了解的不够多也不够深入。或者是即使你了解到的只是片面，不知道他们为什么这么做。你只有整个人切过去的时候才知道那你就可以比较深入的了解他们为什么这么做，这样做有什么好处。然后回过来的时候你就会把这些偷来汇总应用。

现在国内的前端分很多种，像淘宝的前端现阶段可能稍微纯前端一种，像我以前我在有些公司呆过，前端他有写一些后端开发的东西，这些只是一个名字而已，可能跟自己的职者在不同的公司下区分还不一样。

做前端，性能优化是重中之重！

性能优化这块往往是很多刚刚踏入前端开发人员忽略的问题，一个好的前端性能优化对于用户体验也是有着非常大的影响。那么，对于刚刚踏入前端的朋友，嗷嗷也给了一些建议。

嗷嗷认为，对于刚进入前端的同学是比较幸运的，因为现在已经有一些大量的经验，方法，优化等等。对于刚刚入门的时候，认真学习这些东西已经足够了，虽然这些东西比较通用，可能对于某些业务不具备非常好的优化，但对于入门的话用利用这些来了解为什么，比如这些东西为什么这样做会更好，了解这些信息的时候，这样你的能力就会提升了。当你了解这些信息的时候你就会针对业务做一些个性化的定制或者处理。

对于前端性能的优化规范，嗷嗷在这里强调了网站性能优化的 34 条黄金法则，我们称之为雅虎军规，但也不能说是规范，这只是一个建议。但不同的公司都有着不同的背景会导致每个公司都会加上自己的一些特殊的东西。所以让人感觉有很多，每个人出来讲都不一样了。其实最基础的我还是觉得雅虎军规，就是前人的方法总结了，可能由于时代在变，有些方案也会发生变化，所以要自己再去深入研究。

而雅虎现在前端没有以前那么好，就是雅虎辉煌的时候很多人都去看看人家是怎么做的。现在雅虎就没有像以前那么重视了，或者是由于公司的一些原因。我们可能也不是太了解，但是他们曾经留下那些经验是非常好的。

性能和视觉效果，如何解决矛盾？

在前端开发商性能和视觉效果往往是同时存在的，那么鱼雨熊掌能否兼得呢？嗷嗷认为这很正常。比如设计师喜欢用图，因为图能更好的体现出 UI 效果，但是这些图要在客户端呈现的时候，肯定需要下载花费时间的，等待的时候让你觉得性能变差了，但这也不是完成不能解决的。一些方案比如我们现在利用 CSS 技术把几个图合成一个，尽量控制它的请求数。也有一些我们比较推荐的是方案，比如我们用一些新技术，IE6 这种它不支持的，那我们就可以跟设计帅说那我们在好的浏览器中就做出好的效果，低版本的浏览器达到可用的效果就可以，消耗又少很多。这样就尽量找个平衡点，不能这个不能做，那个不能做。

当然还有缓存的合理设置和资源的合并，合理的缓存设置以及……■本文未完，详细部分请参考原文：

http://developer.51cto.com/

一淘清无：Nginx_lua的测试及选择

文/林师授



专家介绍：王晓哲（花名：清无），一淘网技术专家。任职于一淘数据部，负责量子恒道整体技术架构搭建。对海量数据处理、高性能高可用的Web服务相关技术有浓厚兴趣。

对于Web高性能服务器上的选择，这个是很多人头痛的问题。其实Apache、lighttpd、Nginx都用他们优点，在什么情况下我们如何去选择适合自己的Web高性能服务器，如何去搭建一个适合自己的架构环境，这个是一个很麻烦的事情。接下来，在ADC 2012（Alibaba Developer Conference 2012）大会上，51CTO记者有幸采访到了一淘数据平台与产品部技术专家——清无（花名），为我们解读Nginx_lua的一些优势及劣势，以及在高性能服务器上的选择。

首先让我们来了解一下Nginx_lua的设计指导思想：

1、基于Nginx快速开发高性能、大并发的网络服务。

2、提供“同步非阻塞”的I/O访问接口简化I/O多路复用体系中的业务逻辑开发：

◇“同步”的主体是用户代码与其发起的I/O请求处理流程之间的时序关系，意即I/O请求处理完成前用户代码将一直挂起。

◇“非阻塞”的主体是服务进程，意即I/O请求的处理不会导致服务进程阻塞等待，而是可以继续处理其他请求的用户逻辑。

Nginx的特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页伺服器中表现较好。目前中国大陆使用nginx网站用户有：新浪、网易、 腾讯，另外知名的微网志Plurk也使用Nginx。

Nginx服务器及Lua版本的选择

1）Nginx高性能开源WEB服务器的选择

清无是在08年开始接触Nginx服务器的，当时高性能的开源WEB服务器还有lighttpd，那么一淘网技术专家清无为什么会选择Nginx呢？Nginx哪方面比较有它的优势？清无介绍说，lighttpd和Nginx的比较中，有一个很明显的缺点就是lighttpd的模块机制设计的很不好，lighttpd的模块机制过多的把模块本身的请求处理逻辑和底层的网络事件的处理组合在一起，所以不像Nginx的模块结构这么清晰，当然Nginx的模块设计很大程度上也借鉴了Apache的这种模块设计，所以这块有一个先天的优势。当时其实他最早接触lighttpd，然后Nginx出来以后，就对比它

一淘清无：Nginx_lua 的测试及选择 II

们模块结构上的差异后，觉得 Nginx 似乎更有优势一些。实测对于我们这种网络 I/O 密集型的应用来说，只要不是你实现的这个逻辑有多大缺陷，其实在放 lighttpd 或者 Nginx 差别不是特别大。

在比较选择的过程中，首先从架构出发，如果有问题的话无论你实现如何它都是有问题的，所以我的比较首先在架构搭建上，每连接或者每请求单线程单进程这种服务模型，直接就被刷掉，肯定不可能做到很高的服务能力。余下来清一色的都是基于 RO 多路的这种结构体系，那么在这个体系上我们才去检验，实际上拿一个 IPP 的请求来压测看它实现的质量如何，通常来说这部分一旦架构体系决定以后，实测这个性能差异不是特别的大，除非说是某个特性一个实现另一个没实现这种情况，我们测出来的差异通常是在 10%-20% 上下波动而已。

2）Lua 版本的选择

在小编与清无的交流中了解到目前一淘网所使用 lua 的版本是 5.1.2，当小编提出是否版本越高性能越强时，清无则认为不太对。对于 lua 来说每一个版本的变化意味着它将加入新的语法元素或者变更了内部的一些实现的方式。严格意义上并不说明它的性能就好，比如对 5.2 和 5.1 来说，不管对于环境表或者其它的一些机制的修改上面，严格的来说他都是一种新的语言了。所以目前来说迁移到 5.2 最大的障碍其实还是 5.2 里面对于底层接口的这种概念的变化。因为 5.1 里面对于一些方面下了很多工夫，然后使用它的全局表加环境表这种机制。但是 5.2 里面彻底取消了全局表的概念，也取消了 CU 级别上一系列对环境表操作的接口，对我们来说肯定是不能平滑的迁移到 5.2，如果有这个需求的话，我们可以做，但目前还没有看到这个需求。另外一个阻碍我们升级版本号的问题是 Lua JIT，lua JIT 的性能比标准的 lua 要高很多，所以深层里面我们通常用 JIT，但是 luaJIT 目前对 lua5.2 的支持并不是那么紧，它目前还是以 5.1 为主，所以这块我没可能较长的时间跟着 lua JIT 的脚步来。

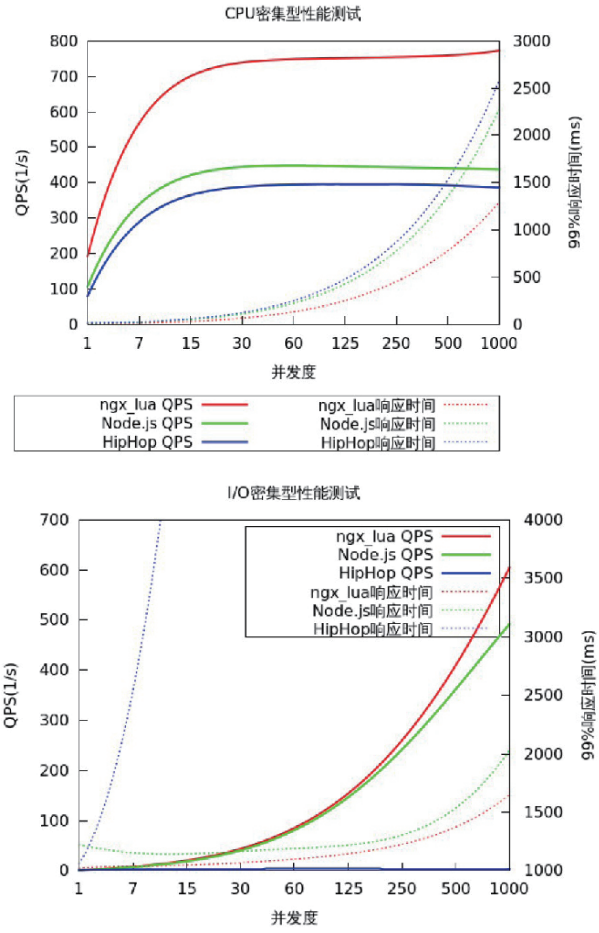
在一淘网的应用中，清无介绍说，Nginx_lua 主要应用在两块地方，一块是传统的一淘数据库量子统计店铺经，数据接口部分完全是用 Nginx_lua 来做。另一块是一淘的广告部门有一部分数据接口也使用着 Nginx_lua。

Nginx_lua 的性能测试比较

其实也有很多人一直还在使用 Nginx_php 这种组合搭配，对于 Nginx_lua 组合的优势在哪里呢？清无介绍说，Nginx+php 之间是要有进程之间通信的，这样以来基础的性能开销就很大。lua 是嵌在 Nginx 进程内部的，它不需要有两套进程在那里独立工作。所以这块从结构上来说就有决定性的优势在里面。再加上线程之间通讯的时候需要大量的反序列化和序列化的工作，然后两套进程带来额外情况是更多的进程更多的切换开销，所以单机上面 Nginx_php 要比 Nginx_lua 要低很多。但是相对来说仍然要回到我们做什么事情上面，因为 Nginx_lua 目前最大的劣势就是周边的模块相当的不健全，我们需要大量的时间来积累这些模块。php 积累了十几年的时间了，如果说你对性能的要求并不是那么高，我的并发数

一淘清无：Nginx_lua 的测试及选择 III

就是几十，那么你用 php 就是最合适的。但是如果像一淘数据的数据接口，机器数就那么一点，因为我的大量成本在 MySQL 集群上面，它是这块的主力，那么对外的数据接口我希望尽可能降成本，并发数又非常大，php 肯定是不行，那么我们就要选择 Nginx_lua。但这块的话对模块的劣势看起来不是那么大，因为它的逻辑相对来说较为固定，我们可以忍受这样的成本，我们去为这个逻辑来定制一些模块。



从上面的两张性能测试图中我们总结 Nginx_lua 的适用场景：

网络 I/O 阻塞时间远高于 CPU 计算占用时间、同时上游资源非瓶颈（可伸缩）的网络应用，如高性能网络中间层、HTTP REST 接口服务等；

期望简化系统架构，让服务向 Nginx 同质化的 Web 站点；

Nginx_lua 的优势和劣势

对于 Nginx_lua 的劣势在刚刚和 Nginx_php 的对比的时候清无也介绍了一个是周边模块不完善，不健全的问题。如果你用到的这个东西比较复杂的时候可能生产力上不去，目前 Nginx_lua 最适合的人员是数据接口层，以及所有的网络中间层，你需要最求并发，高性能的网络中间层。因为它本身的逻辑相对来说比较简单，或者完全用 lua 本身就可以变现出来，这个用起来收效比例是最大的。那么如果你目前要做一个复杂的 WEB 访问站，有大量模板要套，有大量的复杂逻辑嵌在里面，然后要访问 mail 要访问其他服务的话，目前来说我觉得还是 php 或者其他比较成熟的语言。就我们目前应用来说也是这样，中间层会大量的使用 lua，但是前端展现层的话要么全部移到浏览器上面用 JS+ 模板的形式来实现，要么就是用 PHP 这样来做。

另外的劣势就是调试的辅助工具不太多，因为高级点的 php 程序员会往往会使用 XDebug 或者其它的调试工具，可以单步调试，在线调试。跟 php 相比目前还欠缺这样的一个机制。到时候我们会仿照 XDebug 去实现 DPT V2 协议，我们实现兼容 DPT V2 这样的一种机制内连到 Nginx_lua 里面，那样 Nginx_lua 也可以单步调试。到时候我们也会分享给大家。

最后我们来归纳一下清无介绍的几点优势和劣势：

优势：

同步非阻塞 I/O 形式直观易懂，并发服务能力强

CPU、内存运行开销低

■本文未完，详细部分请参考原文：

http://developer.51cto.com/art/201207/350070.htm

编者按

架构师架构师是公司的“金领”，很少需要考虑生存的问题，从而有更多的精力思考关键技术，形成“强者愈强”的良性循环。当然，冰冻三尺非一日之寒，成为一名合格的架构师是一个漫长的积累过程……

合格的架构师是如何炼成的

【51CTO 专稿】架构师是公司的“金领”，很少需要考虑生存的问题，从而有更多的精力思考关键技术，形成“强者愈强”的良性循环。当然，冰冻三尺非一日之寒，成为一名合格的架构师是一个漫长的积累过程。对于任何一个软件开发人员来说，架构师都是一个令人向往的角色。那么程序员们要想如何去成为一名合格的架构师，我们还要先从一名合格的架构师的身上去了解，了解架构师们的一些成长经历，了解他们是如何加强自己的技能修养，了解他们是怎样的一个自我培养的过程等等。下面小编就以上问题采访了新浪云计算架构师从磊老师。



对于任何一个软件开发人员来说，架构师都是一个令人向往的角色。那么从程序员与架构师有没有比较明显的区别呢？从从磊老师的介绍中我们知道，其实架构师和程序员的界限并不是很大，比如现在仍然在每天写代码。其实成长首先来自于自身的学习，而阅读成熟项目的代码会使人受益匪浅，其次就是来自于所从事领域的经验，要了解分布式系统的特点，在做项目时，要能够关注性能、扩展性、可靠性、可用性等指标。

架构师其实就是一个漫长的积累过程，从准备期到动荡期我们都是不停的探索，学习。大部分的架构师的年龄都是在 35 岁左右，这个阶段就是程序员的黄金期。那么我们如何以一个架构师的标准来加强自己的能力呢？下面我们来看看从磊老师的一些建议。

如何去加强技能修养？

程序员要成为一名架构师发展，需要进一步加强技能的修养。从磊认为，对于互联网公司来言，最重要的技能是对网络和分布式系统的理解，网络上面的书籍很多，比较麻烦的是分布式系统，需要结合很多实际的 paper 和方案来理解。对于这些知识，他个人推荐不断的读书，反复的读书，因为同一个知识，在不同的项目经验后，绝对不是不同的理解。

如何去自我培养？

当然要成为一名架构师仅仅通过理论学习是不行的，在从磊的介绍中了解到，其实他理想中的架构师要求很高，不仅有丰富的编码经验，而且还要熟悉硬件性能优化、内核调试、网络故障排查、系统安全、分布式系统设，还有了解国内外技术的新趋势和特点，最重要的是，还要善于与人沟通，敢于排除不同意见，敢于承担责任，了解团队内工程师的特点，善于将他们组成一个整体。本文未完，请阅读原文：

http://developer.51cto.com/art/201208/351656.htm

架构师的成长历程：实践出真知

到架构师，大部分程序员与小编一样都会有一个同样的问题：架构师到底与普通程序员有什么区别？他们平时都在做什么？想什么？也许在大部分的普通程序员的眼里，架构师已经成为了他们唯一的一个目标。

说到架构师，大部分程序员与小编一样都会有一个同样的问题：架构师到底与普通程序员有什么区别？他们平时都在做什么？想什么？也许在大部分的普通程序员的眼里，架构师已经成为了他们唯一的一个目标。也正因为架构师是一门建立在科学，工程基础之上的艺术，一个充满魅力，极其有趣的工作；给人带来了一种不被束缚，不按别人的思路走，自由描绘无限风景的期盼。

架构师是没有终点的，它是一种技术的体现，只有坚持不懈的专研和努力，“强者愈强”这才是一名架构师的价值所在！

如何“强者愈强”？在架构师的成长历程专题的采访中，张巍（Denny）老师给出了他自己的一些看法。



DennyZhang, 架构师，现效力于 shopex 电商云部门

张巍的简短介绍：早先在 EMC 从事分布式文件系统的开发，现致力于构建一个高可用，自维护的电子商务云平台。常年痴迷于 emacs 和 linux，近年来玩玩 erlang。

从选择方向到自我超越

从对 IT 行业感兴趣到开始程序员的生活，Denny 感觉很多 IT 人的职业生涯并不是一开始就有着远大的目标提引着我们向前。它倒是像一个不断跌宕的河流，在碰撞中向前。也因为 Denny 在读大学，读研究生学的都是计算机。自然而然地，慢慢走上 IT 的道路。做着做着，慢慢发现了乐观。特别通过 elisp/emacs 能魔术般地解决一些生活中的问题，由此更激发了他想不断超越自我的想法。

从程序员和架构师当中，Denny 认为，架构师相比软件开发人员来说，主要只是要求前瞻性更好一些，经验更丰富一些而已。两者之间并没有，也并不应该有清晰界线。就像是 linus（Linux 核心的创作者）从来也不会以架构师自居，在他看来更不能妄谈架构师的成长之路。唯一值得提的感触就是：就像是玩超级玛莉一样，每解决一个问题，每受一个搓折，我们的经验值应该要增加的。经验充足了，自然就能预见一些问题，培养一些直觉了。就 Denny 认为，他还有一段……本文未完，请阅读原文： ■

http://developer.51cto.com/art/201208/352283.htm

程序员成熟的标志

程序员在经历了若干年编程工作之后，很想知道自己水平到底如何？自己是否已经成为成熟的程序员？虽然程序员会对自己有一个自我评价，但是，自己的评 价和社会的评价、专业的评价会有差异，所以程序员自己并不能肯定这个评价。现实中，除了各种证书之外，很少有人会专门给出一个程序员的成熟度的评价。人们往往是偶发性地就事论事地对程序员的工作作出好与不好，行与不行的评论。因此，程序员对此感到很茫然，不知道要从那些方面去评价自己的能力。

一个程序员到底成熟不成熟，我想从以下几个方面谈谈自己的看法。

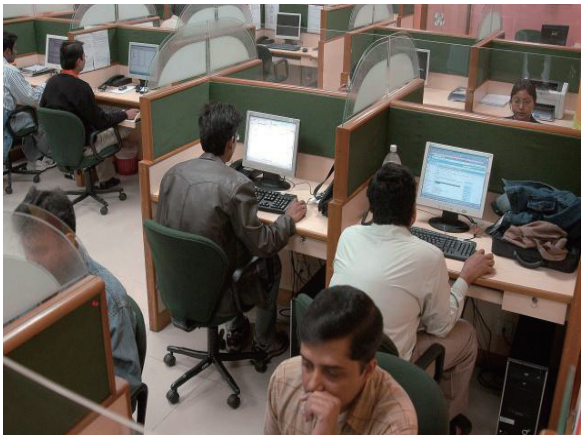
1、技术标志

如果程序员不会编程序那决不是程序员，程序员至少要掌握一门程序设计语言，要能够用这种语言去编写程序去解决他想解决的问题。但是，成熟的程序员往往掌握不止一种程序语言，三到四种语言的掌握是必须的，一种二种语言的精通也是必须的。

除了从掌握程序设计语言个数之外，我们还可以从其他几个方面去看看程序员在技术上水平。例如，函数编写能力（命名、格式、大小、分类、参数、复用等），面 向过程的能力，面向对象的能力，数据库技术能力，效率处理能力，安全处理能力，网络处理能力，软件构架能力，人机交互能力，通用软件能力，软件文档能力等 等。尤其是面向对象技术的掌握和运用，以及面向服务的技术都是成熟程序员所必需掌握的。

2、时间标志

虽然程序员的天资、素质、基础知识各不相同，所经历的工作内容以及环境也不相同，但是，时间也是程序员成熟程度的标志之一。一般程序员需要经过三到五年的 时间才能日趋成熟。其中入门需要一年，成长需要两年。这是我经过长期观察得到平均数据。我并不认为成熟时间越短程序员就越聪明，就越了不起。享受每个阶段 充分的时间，会让自己成长更加充实、更加成熟。当然，也有超期而不成熟的情况，这也是很正常的。



3、项目标志

程序员的社会性是程序员成熟的标志之一。没有参加过项目的程序员，程序编得再好，只能是纯程序类的程序员，是一个孤独的高手，是一种个人型的程序员，远没 有成熟。项目作为社会性活动，体现了项目的社会价值。所以项目能力也是程序员成熟的重要标志之一：项目能力包括参加项目的个数、项目的大小、在项目中承担 的角色等等。就项目承担的角色而言，主持开发（项目经理）3 个以上项目是必须的，这是一个必要条件。

程序员成熟的标志 II

一个程序员如果没有主持过开发，无论参加过多少项目的 开发，无论是在程序编写或项目设计上发挥了多大的作用，是很难被称之为成熟的，因为项目的组织、协调和管理是反映一个程序员成熟程度的又一个标志。就如同 一个程序员能参与过 10 个以上大大小小的项目或能参加或能主持两个以上大型项目的开发，其成熟程度是可以信赖的。若低于此数，则说明程序员离成熟还有相当 的空间。“我们在项目中成长” 可见项目对于程序员的意义是多么的巨大。

另外，一般程序员只是为一个企业客户进行开发一个或多个项目，或同行业的企开发项目，如果程序员能够如果程序员能够开发过多个行业的项目，其成熟度要比一般人要高一些。

4、思维标志

幼稚和成熟在思维方式上还是有很明显的区别的。就程序员而言，不成熟的程序员逻辑性不强，程序编得没有条理，即使程序员自己进行了解释也没人能看懂。而成 熟的程序员应该具有很强的逻辑性，程序编得井井有条，不用解释别人也能看得懂。这种逻辑性还体现在软件的构架设计、数据库设计、算法设计等多个方面。程序 员通过全集子集概念、时间概念、顺序概念、重点非重点概念等对各种事物进行逻辑分析。例如，以顺序概念为例，不成熟的程序员往往会采用自底向上的思维方式 来开发程序。他们先考虑程序的具体实现，然后再考虑功能设计、最后考虑构架设计。而成熟的程序员则采用自顶向下思维方式，先考虑构架设计、再考虑功能设 计、最后才考虑编程的具体实现。前者思维方式主要是出于工作惯性，只适合入门阶段，而后者思维方式反映了后者的进步，适用于各种项目开发或大型项目的开发。

除了在思维内容上的逻辑性之外，程序员还应该处理好动脑和动手的关系。重视思维本身就是一种成熟的标志。成熟的程序员的思考时间要大于动手编程时间，想好 之后只要一次就编程成功，而不成熟的程序员往往动手编程时间要远大于思考时间，而且是边做边想，通过反复来逼近最终目标。

另外，在思维范围上，成熟的程序员要比普通的程序员有更开放视野。他们更容易去接受新的东西，更容易不受各种约束去考虑问题，更勇于去挑战自己和高手。

5、与人交往

很多人认为程序员是和计算机打交道的行业。这只是这个职业的特点。但是，只要是工作必然就是一种社会劳动。而社会劳动则必须和人进行交流和沟通。尽管程序员的劳动工具是计算机，但并不意味着程序员只想着这个工具。从这个工具的下游来看，程序员还是要考虑用这个劳动工具生产出来的软件产品是否有人购 买，是否有人使用，是否运行正常，从这个工具的上游来看，是谁让程序员了解设计方案的，是谁让程序员编程序的，是谁让程序员程序通过验收的等等。因此程序 员在软件制作各个环节都会与其他人打交道。只有和人进行有效的交流和沟通我们的工作才能进行下去才能做的更好。

（本文未完，请阅读原文）■

http://developer.51cto.com/art/201207/349025.htm

身居中国的法国程序员谈Python/PHP

大家好，作为旅居北京的法国人 Guillaume Bouriez，我跟各位一样每天生活在街巷胡同之间。我目前在豆瓣公司工作，还有个刚刚三岁的小儿子。我是个不折不扣的 Python 粉儿，并且乐于到处宣扬这一点。并且上一份工作让我对 PHP 也是非常熟悉。

网络以及移动领域的方方面面都非常吸引我，而且我很关注中国在此时此刻正悄然发生着哪些变化。我是客服工程师出身，拿过硕士文凭，汉语学习则是在四川大学完成的。另外我对中国古代文化很感兴趣，也读过不少这方面的著作。

Guillaume Bouriez 是一位来自法国的开发者，目前在国内非常小资的社区豆瓣工作。谈到豆瓣，鬼庸认为："豆瓣称得上当今中国最具个性魅力的网络公司。

看看我们所推出的产品，再与其它竞争对手加以对比，大家应该会明显感到豆瓣所打造的使用界面简洁而直观，同时拥有高质量的信息内容。正是这一点令我决定成为该公司的一员。我真心希望自己参与的项目及产品能够为用户带来乐趣，这也正是我在收到豆瓣公司的录用通知后欣然接受的原因。"

在交流的过程中，鬼庸说："我是个不折不扣的 Python 粉儿，并且乐于到处宣扬这一点"。

鬼庸还告诉记者："我的上一份工作是在北京一家规模不大的法国公司担任 CTO，当时我们的技术堆栈主要是 LAMP，而其中 P 指的是 PHP，我其实对这两种语言都很熟悉。"

Python 与 PHP 之间的差异

在鬼庸看来 Python 是一款拥有健全基础的优秀工具；而相比之下 PHP 在基础方面则很不健全甚至干脆谈不上什么基础，这就注定了其不怎么好用的命运。可以说再多的补丁也没法彻底解决 PHP 的问题，它从根本层面上已经不可救药了。其实大家可以把 Python 和 PHP 间的区别与 git 跟 svn 联系起来，这两对冤家的情况非常类似：git 与 svn 都能达到我们的预期目的，但毫无疑问 git 在架构方面具有明显的压倒性优势。

也许有人会以 Facebook 以及维基百科作为反例，但情况恰恰相反：表面上来看，二者的创建方式似乎证明在没有选择的情况下，使用 PHP 同样能够打造出精彩绝伦的产品。但很显然，如果 Facebook 或者维基百科的工程师们有重新编写代码库的机会的话，他们肯定会毫不犹豫地投向 Python 的怀抱。因此，在我看到很多中国的新兴企业使用 PHP 开发产品时，总会感到一种莫名的伤感——这绝对是个悲剧性的错误。如果大家仍然对我的观点有所怀疑，那么不妨想想看近期推出的新产品中有哪一款使用了 PHP：Quora？Github？ Path 又或者 Instagram？现实告诉我们，没人再用 PHP 了。

我还听到这样一种论调，有人认为 PHP 的开发人员群体比 Python 更庞大（同样也比 Ruby 和 node.js 更大），因此在招聘人才时也会轻松得多。这其实根本不是问题：既然大家希望能够吸纳优秀的开发人员为自己服务，那么所谓优秀就是要

一个身居中国的法国程序员谈 Python 与 PHP II

能快速从一种开发方式转移到另一种。Python 无论在学习还是使用方面都比 PHP 容易得多，我在法国的一位好朋友就利用某个百无聊赖的暴雨周末教会了自己的小孩用 Python……

做开发需要挑选一件趁手的兵器

当大家只懂得使用 PHP 时，当然会以 PHP 的方式构思自己的解决方案，并顺理成章地认为 PHP 是解决一切需求的最佳方案。但随着掌握的语言越来越多，我们会发现自己的思维方式在不断扩展，并能够在面对某些问题时拥有更多更有竞争力的解决办法。

如果大家奉行的处世哲学是“不谈意义、只求结果”，那么 Windows 系统中的记事本功能可能常常会作为各位编辑文件的首选方式，对吧？当然，用记事本编辑文件倒也不是不行，不过这么干真的既不专业、也不实用。我跟自己的大多数同事一样用 Vim 解决这类问题，而且我也没听说过哪位专业开发人员会乐于通过记事本搞定一切。这在选择编程语言方面同样说得通：一位优秀的开发人员必须拥有应对多种语言的能力，就算是相对较差的语言也需要掌握。但如果他真的热爱工作、乐于提升自己的技能水平进而写出更令人满意的代码，那么我真的无法想象他会认为每种语言在实际应用中的表现都差不多。有些好、有些差，这就是现实。

除此之外，让自己拥有不同于其他竞争对手的特色也是十分必要的。也就是说，如果大家都在用 PHP 或者 .NET，那么我们不妨早点开始学习 Haskell、node.js 或者 Python，即使从目前看来这些技能对我们的职业规划还没啥具体影响。在豆瓣工作的这段时间我非常快乐，因为我身边的同事都具有强烈的进取精神。这种精神的实际表现就在于他们喜欢尝试不同类型的事物，希望在更多选择中找出最佳方案。这样的积极气氛非常难得，我感到乐在其中。■



Guillaume Bouriez 中文名：鬼庸

豆瓣：ruim

51CTO 精品杂志推荐《Linux 运维趋势》

《Linux 运维趋势》是由 51CTO 系统频道策划、针对 Linux/Unix 系统运维人员的一份电子杂志，内容从基础的技巧心得、实际操作案例到中、高端的运维技术趋势与理念等均有覆盖。



本杂志长期处于探索期，需要更多来自大家的意见与参与。

如果您对这份电子杂志感兴趣，那就请您下载阅读；想要帮助我们做的更好，请进入我们的 Linux 运维趋势技术圈留下您的宝贵意见建议。

读者讨论组：http://g.51cto.com/linuxops/

大数据时代 数据分析师的尴尬

这个七月着实忙碌了好久，也是工作的小事情不断，回头想想自己忙什么了，感觉好像真的没什么忙的，不过却又忙碌了这些天。



这几天和一个圈里的好友聊天，问我怎么来看待现在的工作状态。她也是一个做网游数据分析的分析师，她说一天的工作除了固定的发完每天要做的那部分，似乎剩下的时间就没了什么事可做，最感觉有点后怕的是，作为一个 leader，下面一群人还需要指点和安排。这种空洞让她有些不能心安。

事实上，我有很长的一段时间也是这种状态，最后练就的本事时每天的必须工作只需要半小时就搞定了，剩下 7 个多小时就闲着了，闲的心里面有点害怕，因为怕这种闲。而造成这种感觉其实有时候感觉自己是在浪费自己的青春和时间，想努力抓住一些东西，但是又无法抓住。网游数据分析师说来这个词曝光率都不是很高，上有数据挖掘工程师的大帽子，之后又是业务分析师，运营团队的人，又是研发策划的人压着，其实有时候感觉挺苦逼的，挺悲催的地位，外加上环境和目前的分为并没有看重这个行业进步和发展，也就这样了。

现在几乎每天看到 Big Data，数据分析这些热词，说实话我对于 hadoop，mapreduce 这些不是很关心，因为我不是一个要去做技术的人，我对于经济学，心理学，营销学也不是很关心，因为我也不是一个要去做管理的，纯粹的运营的人，我有自己一个独立的称号网游数据分析师。

也许这个职位的价值不被数据挖掘工程师认可，因为你的技术没他们好，连算法都搞不懂；

也许这个职位的价值不被运营人员所认可，因为你的分析和知识他们就能做到，连业务都没吃透；

也许这个职位的价值不被研发的人所认可，因为你根本就不懂研发，狗屁不是。

但是为什么还要坚持呢？因为存在价值。

因此就不必怀疑自己自己的价值，如果你热爱这份职业。

一个网游数据分析师不该停留在那些指标上，侃侃而谈就 ok 了，我们要去吃透那些指标，我们要去理解业务，驾驭上层的数据。同时，我们却又要不断的去探究为什么，因为数据挖掘工程师不会告诉你为什么，他们只能告诉你 what，而你作为一个数据分析师，在业务者与挖掘者之间，就要解决，最终服务于运营业务的 how。

所以我们可以不懂得高深的算法，但我们要懂得如何将算法应用，如何驾驭那些软件。我们不懂得市场，但是我们要懂得一点长尾，懂得一点怪诞心理，懂得一点社会性。我们不懂得设计，但是我们懂得一点用户体验，购买决策。

其实思考了很久，我觉得作为一个数据分析师，最终就是在构建个非常完整和健康的 CRM。基本上层和底层的数据构建和服务对象都是 CRM，（本文未完，请阅读原文）■

http://database.51cto.com/art/201208/351851.htm

数据分析师的级别

1、数据跟踪员

虽然这个工作的人还不能称作数据分析师，但是往往作这样工作的人还都自称是数据分析师，

这样的人，只能通过 ××× 系统看到有限的数据，并且很少去处理数据，甚至不理解数据的由来和含义，只是机械的把自己看到的数据拷贝出来，转发给相应的人。

这类人发出来的数据，是否有意义，怎么解读，他自己不知道的，只能期望收到数据的人了。

2、数据查询员 / 处理员

这些人可以称为分析师了，他们已经对数据有一定的理解了，对于大部分数据，他们也知道数据的定义，并且可以通过监控系统或者原始的数据，处理得到这些数据。

统计学的方法，这批人还是很精通的，统计学的工具，他们也是用起来得心应手，你让他们做一下因子分析，聚类肯定是没问题，各类检验也是用的炉火纯青。

他们的不足是：1、如果不告诉他们命题，那么他们就不知道该应用什么样的方法去得到结论了。2、对于数据的处理没问题，但是却没有一个很好的数据解读能力。只能在统计学的角度上解释数据。

3、数据分析师

数据分析师这群人，对于数据的处理已经不是问题了，他们的重点已经转化到怎么样去解读数据了，同样的数据，在不同人的眼中有不一致的内容。

好的数据分析师，是能通过数据找到问题，准确的定位问题，准确的找到问题产生的原因，为下一不的改进，找到机会点的人。

往往科班出身的人，欠缺的不是在处理数据上，而是在解读数据上，至于将数据和产品结合到一起，则是其更缺少的能力了。

4、数据应用师

数据应用，这个词很少被提到。但是应用数据被提的很多，分析了大量的数据，除了能找到问题以外，还有很多数据可以还原到产品中，为产品所用。

典型的是在电子商务的网站中，用户的购买数据，查看数据和操作的记录，往往是为其推荐新商品的好起点，而数据应用师就是要通过自己的分析，给相应的产品人员一个应该推荐什么产品，购买的可能性会最大的一个结论。

国内能做到这个级别的数据人员还真是少的可怜，甚至大部分人员连数据的视图都搞不定，而真正意义上的能数据应用师，可以用数据让一个产品变得更加的简单高校。

5、数据规划师

数据规划师，不能说水品上比数据应用师高多少，而是另外一个让数据有价值的方向。

往往在实际的应用中，数据都是有其生命周期的，用来分析，应用的数据也是，这点上，尤其是在互联网公司更加明显，一个版本的更新，可能导致之前的所有数据都一定程度的失效。

数据规划师在一个产品设计之前，就已经分析到了，这个产品应该记录什么样的数据，这些数据能跟踪什么问题，哪些记录到的数据应该用到数据中去，可以对产品产生什么样的价值。■

编者按

什么是整洁代码？不同的人会站在不同的角度阐述不同的说法。“整洁的代码简单直接。整洁的代码如同优美的散文。整洁的代码从不隐藏设计者的意图，充满了干净利落的抽象和直截了当的控制语句。”

代码整洁之道

现在的软件系统开发难度主要在于其复杂度和规模，客户需求也不再像 Winston Royce 瀑布模型期望那样在系统编码前完成所有的设计满足用户软件需求。在这个信息爆炸技术日新月异的时代，需求总是在不停的变化，随之在 2001 年业界 17 位大牛聚集在美国犹他州的滑雪胜地雪鸟（Snowbird）雪场，提出了“Agile”（敏捷）软件开发价值观，并在他们的努力推动下，开始在业界流行起来。在《代码整洁之道》（Clean Code），提出一种软件质量，可持续开发不仅在于项目架构设计，还与代码质量密切相关，代码的整洁度和质量成正比，一份整洁的代码在质量上是可靠的，为团队开发，后期维护，重构奠定了良好的基础。在这本书中作者提出了注重实际开发实践的细节，而不是站在空洞的理论来谈论整洁之道。

什么是整洁代码？不同的人会站在不同的角度阐述不同的说法。而我最喜欢的是 Grady Booch（《面向对象分析与设计》作者）阐述：

“整洁的代码简单直接。整洁的代码如同优美的散文。整洁的代码从不隐藏设计者的意图，充满了干净利落的抽象和直截了当的控制语句。”

整洁的代码就是一种简约（简单而不过于太简单）的设计，阅读代码的人能很清晰的明白这里在干什么，而不是隐涩难懂，整洁的代码读起来让人感觉到就像阅读散文 – 艺术的沉淀，作者是精心在意缔造出来。

一：命名

命名包括变量、函数、参数，类等，一个好的命名能够很好的表述其所承载的业务，从命名上就已经很好的答复了为什么存在，做了什么事，应该怎么用等的大部分的问题，阅读者看到它的时候不必去深究其实现细节，一切都在命名上一目了然。一个好的命名必须是名副其实，不存在歧义（双关语或常见属于冲突），直接了当（否定语句或者误导性命名）。

二：函数：

从汇编 /C 时代开始的到现在函数一直都存在与我们开发中不可或缺的一部分，结构化组织，重用 . 作为函数式语言的一等公民，所有程序的第一组代码。

好的函数必须足够的小，其次还是足够的小。很容易想像阅读上千行的代码，是多么巨大的自我心理挑战，在实习的时候工作于毫无分层逻辑的 WinForm 平台下，完全依赖 RAD 模式带来后置 cs 页面上千行的代码，每次修改都令我恼怒，恨不得重写整个业务逻辑。

一个函数在于短小精悍，只作一件事情，并做好这件事，只做一件事才能得到更好的利用函数名表述自己。

好的函数还应该是 CQS（查询命令分离）无副作用的（不存在隐藏歧义的背后逻辑），并对其他类型不存在“依恋情节 (Feature Envy)“（类中

代码整洁之道 II

的变量被所有的函数使用这是理想的高内聚，万物皆有其位，而后物尽归其位）。

函数的参数应该足够的少，无最好，一次之，再次为二，尽量避免三个以及三个以上，对于太多的参数你可能该采用 IntroduceParameterObject（引入参数对象）。

重复的代码。重复在软件系统是万恶的，我们熟悉的分离关注点，面向对象，设计原则…都是为了减少重复提高重用，Don't repeat yourself！(DRY)。

三：注释、格式：

并不是写出完备的注释就是好的开发人员，如果代码清晰的表述自己意图，那么注释反而多余。在《重构－改善现有代码之道》中 Martin Fowler 指出多余的注释是一种代码坏味道。就是好的注释随着项目的维护不断的重构很多时候也会变得不那么适应，而我们很少会去主动维护。再则误导性的注释更为使用者所憎恨。当然有时我们也得使用注释，注释并不是万恶的，当我们没法用代码来描述自己的时候，我们需要注释去描述意图；多余有副作用的代码给使用者提供警告注释。

TODO 开发时进度控制，比如你在进行较大规模领域重构，目前有些逻辑不再适应，不那么自然，而对它的重构还在任务列表最后，你可以选择标注在 TODO 中，最后完成从 ToDoList 中去掉每一个 TODO 任务。

良好的代码格式，会使得我们阅读更容易，一套共同的格式会让我们查找理解更快速。每个团队都应该遵循一套固定代码格式规范，整个软件系统的统风格统一，而不是各自为政各成一。

四：对象和数据结构：

数据结构指的就是数据的载体，暴露数据，而几乎没有有意义的行为的贫血类。最常见的应用在分布式服务，以 wcf，webservice，reset 之类的分布式服务中不可或缺的数据传输对象（DTO）模式，DTO（Request/Response）就是一个很典型的数据载体，只存在简单的 get，set 属性，并且更倾向于作为值对象存在。而对象则刚好相反作为面向对象的产物，必须封装隐藏数据，而暴露出行为接口，DDD 中领域模型倾向于对象不仅在数据更多暴露行为操作自己或者关联状态。

数据结构和对象之间看是细微的差别却导致了不同的本质区别：使用数据结构的代码便于在不改动现在数据结构的前提下添加新的行为（函数），面向对象代码则便于不改动现 有函数的前提下添加新的类。换句话说就是数据结构难以添加新的的数据类型，因为需要改动所有函数，面向对象的代码则难以添加新的函数，因为需要修改所有的类。在任何一个复杂的系统都会同时存在数据结构和对象，我们需要判断的是我们需要是需要添加的新的数据类型还是新的行为函数。

隐藏作为面向对象主要特性中的最重要特性，封装隐藏是面向对象中最重要的特性，一个好的面向对象代码肯定是对对象的内部细节做到很好的隐藏封装，封装过后才有是多态，委派之类的。一个好的面向对象的代码一定是具有很好的隐藏封装，易于测试，不稳定因素往往集中在一处很小或者固定的位置，不稳定因素的变更不会导致更大面积的修改扩散。■

本文未完，请阅读原文：
http://developer.51cto.com/art/201208/351293.

你的代码值多少钱？你算过吗

我们规定程序员书写、重构、改 bug 产生的变化行数都算做生产行（Line），把所有生产行分为三个等级：框架（structure）、功能（feature）、逻辑（logic）；

老板雇佣程序员来写代码，程序员的薪水是成本，产生的有效代码是产品。或者说老板从程序员那里买代码。程序员每写一行代码究竟能赚多少钱？不妨来推导一下。

我们规定程序员书写、重构、改 bug 产生的变化行数都算做生产行（Line），把所有生产行分为三个等级：框架（structure）、功能（feature）、逻辑（logic）；将程序员在单位时间内写的三类生产行分别设为 Ls、Lf、Ll；每种生产行的权值分别为 Ws、Wf、Wl。生产行失效率为 P（0 到 1 之间），失效的生产行指在老代码运行过若干次后被重构成了新的代码（在开发阶段快速迭代重构只计算最终的生产行不算失效）或者被验证有 bug 后改正的代码。则该程序员在单位时间内写的代码的价值为 Vt = (Ls × Ws + Lf × Wf + Ll × Wl)；有效代码价值 Ve = Vt × (1 − P)。设该程序员单位时间内的薪水为 S，进一步可以得到该程序员的性价比 C = Ve / S。

公式只有这些，接下来随便假设一些数值代入公式做下计算。设有两个程序员：a（高级）、b（初级），假定 a 月薪 5000，日代码价值 50，（按 22 工作日计算，下同）月代码价值 Vt(a) = 50 × 22 = 1100，则 a 的性价比为 C(a) = Vt(a) × (1 − P(a)) / S(a) = 1100 × (1 − P(a)) / 5000。同理可得 b 的性价比 C(b) = Vt(b) × (1 − P(b)) / S(b)。考虑理想情况，a、b 的代码失效率都为 0 即 P(a) = P(b) = 0，若要 a、b 程序员的性价比相同则有 C(a) = 1100 / 5000 = Vt(b) / S(b) = C(b)；如果 b 的日有效代码价值 25，月有效代码价值 25 × 22 = 550，月薪 2500，则刚好 C(a) = C(b)，公司的薪资结构是平衡的。下面说一些你们可能有的问题和一些引申想法。

“我看到一些公司拿钱多职位高的写的代码量很少，贡献代码最多的是一些新人，性价比怎么会平衡？”代码行数不直接代表代码价值，这也是三种代码权值 W 存在的意义，写框架性的代码靠深厚的功底。框架、功能、逻辑的权值是依次递减的即 Ws > Wf > Wl。项目初期高级程序员基本全部精力都在写底层框架而几乎不写上层应用，往往是每天想 4 个小时写 2 个小时再花 2 个小时重构，项目中后期在框架实在无事可做时高级程序员又往往在写功能、逻辑时贡献的代码量又是最多的。每天框架代码的生产量可能是很低的但是含金量很高，是整个项目存活与维持的基础。

“高级与初级程序员的性价比真的很理想化的相等吗？”不。理想化的性价比比在实验室里创建高温超导体还困难重重。原因在于很多不确定和随机因素。任何程序员都不可能一辈子一直写出滴水不漏不需要返工的高质量代码。上面的计算已经是把代码失效率当成理想的 0 来算的了，如果高级程序员更了解代码中的陷阱与欺骗从而能更好的避开，那就能写出失效率更低的

你的代码值多少钱，你算过吗？ II

代码，也就是说出错率高的新人要写更多的代码才能像高级程序员一样划算。



“行业薪资标准该怎么算？”在求职、入职与人事部门谈论薪资待遇时他们都会说：按掌握的行业统计数据给你的薪资为 XXXX。这句话表面意思我非常理解，即行业薪资标准的建立需要大量统计数据。但我不知道各个公司是怎么统计与处理数据的。我想这么算一下：需要统计两样数据，程序员有效代码价值 Ve，薪资数目 S，这两样数据计算得到的就是上面推导过的性价比 C = Ve / S。当有大量数据时我们可以得到性价比的一个行业平均值 Cavg（如需做其它分析还要得到最大值、最小值、方差等）。在对一个程序员笔试面试过后可对其生产有效代码价值的能力做个初步评估，在此基础上可以给出该程序员的薪水，如果老板开给公司里大多数程序员的薪水过低，性价比高于 Cavg 则可能发生频繁跳槽与被挖角现象；如果开的薪水很高性价比过低，那也许能笼络到大量人才，但是那老板真有钱，还很阔绰。想稳稳当当就构建和 Cavg 持平的性价比即可。

“那身为老板，我为什么不全都雇佣性价比最好的高级程序员？”不可能也没好处。从业人员不可能每个人都有很到位的技术火候。如果老板雇到的全是最顶尖的程序大师却又安排不好这些人员那沟通与管理成本就会上升很多。一个高级程序员带队几个沟通顺畅的人（不管高级还是初级）那么做起事来会很有效率，如果是几个固执的高级程序员聚在一起吵来吵去那么生产效率还不如两个平庸的程序员加一个好的指导思想。此文只说和上面的公式相关的问题，公司内部的管理平衡本文不谈。

“怎样节省成本？”我一向吹嘘的两个程序开发原则是复杂度隔离和代码复用。降低了复杂度，代码阅读修改会变容易，扩展维护成本会变低；书写并使用能充分复用的代码也会减少 repeat yourself 的机会。发展与使用开源软件也是个好的方法。如果一个管理者成天摧程序团队把某某某功能做出来，那真的该反思一下这种只要尽快看到短期结果不求深谋远虑的施压造成的是什么情况：程序员赶工，效果是快速出来了，但代码经过推敲的时间少了，复杂性可能高了，潜藏的 bug 可能多了，以后复用的机会非常渺茫了……长远来讲，不如平心静气鼓励程序员写出高质量的可复用代码，使用、参与、投入几个开源项目，甚至鼓励程序员用一定自由时间做自己想写的程序（Google 有此一招，每周一天，其若干商业项目起源于员工个人项目），这样公司的成本更低。假设一套代码（自己员工写的可复用代码或现成开源项目）有 10 万有效代码价值，还是按一个高级程序员 1100 月代码价值来算，在不出错的理想情况下这套代码可能为老板节省 100000 / 1100 = 90.91 人月的成本，5 个高级程序员也要干 18.19 个月才能做出来，而老板能节省 99.91 × 5000 = 45.45 万元的金钱成本，又省时又省钱，何乐而不为？（本文未完，请阅读原文）■

http://developer.51cto.com/art/201208/351336.htm

国外程序员是如何准备面试的

下面有张问题表，虽然很多问题看似比较低级，但相信我，看完这些问题后，你应当反思自己。即便面试时没有碰到类似这张列表上的问题，你在面试也可以用上已准备好的答案。

一、开门见山

一般来说，你（求职者）在面试中会花很多时间谈论以前的经历。如果你还没有想出你的最具代表性的经历 / 案例，那么你现在是时候该做了。下面有张问题表，虽然很多问题看似比较低级，但相信我，看完这些问题后，你应当反思自己。即便面试时没有碰到类似这张列表上的问题，你在面试也可以用上已准备好的答案。

1. 说说你自己吧

2. 你的短期目标是什么？你 2 到 5 年内的目标是什么？

3. 你的愿景 / 使命是什么？

4. 你想从这个岗位中学到什么？

5. 你为什么认为自己能胜任这个岗位？

6. 除了这个岗位，你还在找哪些岗位？

7. 你有什么样的管理或领导经验？

8. 你有什么样的团队工作经验？

9. 你最满意 / 不满意的经历是什么？

10. 你的强项 / 弱项是什么？

11. 你最擅长处理哪一类的问题？

12. 你是如何减压？如何平衡工作和生活？

13. 如果他人提出的要求有违你的道德准则或商业道德，你怎么处理？

14. 你上次试图向他人出售点子有什么样的结果？

15. 你为什么要应聘我们公司？你对我们公司了解多少？

16. 你认为加入我们公司有什么样的利弊？

17. 你最看重雇主的哪一方面？

18. 你过去遇到的主管有哪些共同点？

19. 如果要在竞争中保持领先，你认为我们公司的员工最需要哪些品质？

20. 你最喜欢 / 不喜欢什么课程？为什么？

21. 你的兼职 / 暑假 / 实习经历中有什么收获？

22. 你有什么样的深造计划？

23. 为什么你的成绩这么低？

24. 你是如何打发空余时间的？

25. 如果我让你的朋友们描述你，你觉得他们会怎么说呢？

26. 什么事最让你沮丧？

27. 你上次在工作中遇到恼火的事是什么时候？结果如何？

28. 你如何提高你的整体效益？

29. 去年你遇到最艰难的决定是什么？为什么它很艰难？

30. 你为什么到现在还没有找到工作？

31. 你在哪些方面没有任何经验？比如：销售、筹款或者簿记。

32. 我们为什么要雇用你？

国外程序员是如何准备面试的 II

上面这个列表中的问题需要深思和反省才能回答，所以在下一次面试之前很有必要提前想一想。有些问题可以有相同的答案。如果问题答案需要讲述一段故事 / 经历，你可以按如下方式来做：

1. 20 秒描述情况。“代码出问题了，整个团队必须找出原因。”

2. 30 秒描述你做了什么。“我认为是内存溢出，我注意到 AbstractMemberCreationFactory 本应该只有一个实例，但它有很多实例”。

3. 20 秒描述结果。“我用一行代码解决溢出问题，我们的产品如期上线。后来，我都会先测试确保不会再有类似问题”。

在每次面试前，浏览整个列表并大声练习你的回答。这样做让你在面试中更加轻松自如，你就比其他应聘者有更多的优势。比如，当在回答问题时，其他应聘者可能会先看着天花板说“嗯”，然后开始回忆过去 5 年中发生的一切。而你则可以微笑地看着面试官的眼睛，再陈述你的答案。

二、事先练习最常用的编程算法，以建立自信

钢琴家在达到下一级水平之前，他们必须先学习一套短曲。这些短曲虽然不曾在聚会上弹奏，但他们要练习特定的部分，比如右手小指和切分音。同理，尽管有些算法你或许没有在你的代码中用过，但它们经常在编程面试中出现。比如：

1. 反转单链表；

2. 反转字符串或反转段落中的单词顺序；

3. 利用星号输出画一个任意大小的圆；

4. 整数转换为字符串，数字型的字符串转换为整数；

5. 写一个函数，返回任意整数对应二进制形式中的“1”的个数；

6. 写一个函数，输出字符串中字符的所有排序。（比如：abc acb bac bca cab cba）

一般来说，可以先不考虑算法的运行时间，都是先用最简单有效算法。然后再尽量减少运行时间。能让算法正常跑起来，总比你花大量时间想一步到位要好。（一步到位的想法虽好，但大部分时候都是事倍功半。）

提示：做练习时，别自欺欺人看答案。

我第一次写“反转单链表”算法是利用课间时间。我并不急，从运行时间较长的方案到优化的方案，我用了半个小时。在验证算法正确无误后，我非常高兴！我知道不看答案我也能解决问题了。（建立自信了！）在面试中，当遇到从未见过的问题时，我以前的经验给了我信心。

三、练习解决问题能力

有些应聘者认为能解决脑筋急转弯问题就有良好的编程技能。如果你也这么认为，那你应当培养猜谜这方面的兴趣和技巧。你可以去图书馆或在网上找这方面的书。然后，挑一些有趣的问题，先别看解答。如果半个小时之内还想不出答案的，才可以查阅解答。

第四、展现出真正的热情

另一个技巧就是要把真正的热情展现出来。作为人，我们应该真诚和热情与他人共处工作。

（本文未完，请阅读原文）■

http://developer.51cto.com/art/201207/348972.htm

全世界有多少Java开发者?

我要这些数字是为了计算内存泄露检测工具潜在的市场。但这些数字困扰了我多年,似乎这些数字背后没有可信的调查和依据,都是拍脑袋想出来的。所以我决定尝试去获得真实的数据。

Oracle 说世界上有 900 万 Java 程序员，Wikipedia 说是 1000 万,而 NumberOf.net 的哥们儿说的很精确:世界上有九百万七千三百四十六个 Java 程序员。

我要这些数字是为了计算内存泄露检测工具潜在的市场。但这些数字困扰了我多年,似乎这些数字背后没有可信的调查和依据,都是拍脑袋想出来的。所以我决定尝试去获得真实的数据。

这确实是一个很有挑战性的工作。地球上有 70 亿人口,我不可能一个一个去问。所以必须得想点别的办法。

通过阅读和观察不同的信息来源,我决定深挖四个来源:

各个政府发布的劳动力报告

编程语言排名报告,例如 Tiobe 和 Langpop

找工作门户 Indeed.com 和 Mondter.com

流行 Java 工具的下载量,例如 Ecilpse 和 Tomcat

我打算通过这些信息计算出一个比较合理的数字。

世界上一共有多少程序员?

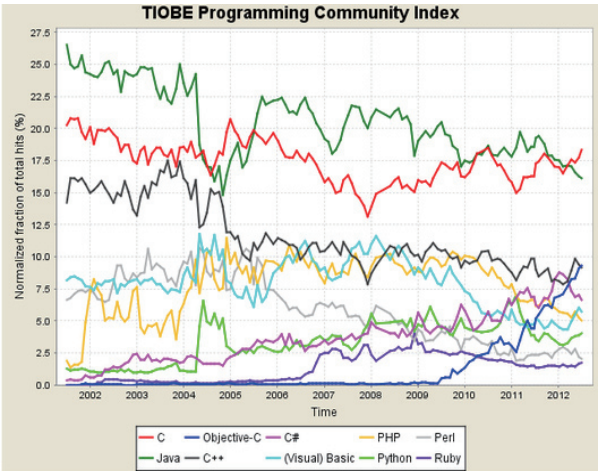
世界上大约有 70 亿人口。除去 sub-Saharan Africa 和 rural Asia 这些不大可能有软件产业的地方以外,我们可以计算的人口大约有 50 亿。

在这 50 亿中,有多少人在开发软件呢? StackExchange 上的一个答案给我们找出不同国家软件开发者的比例指了一条明路。通过 US, Japan, Canada, EU, UK 这些地区作为样本,我们可以估算大约 0.86% 的人口是以软件开发为职业的:

Country	Population	Developers	%
Canada	33,476,688	387,000	1.16%
EU27	502,486,499	5,900,000	1.17%
Japan	127,799,000	1,016,929	0.80%
UK	63,162,000	333,000	0.53%
US	313,931,000	1,336,300	0.43%
		Weighted average:	0.86%

编程语言的流行度

在这个环节,我们将使用 TIOBE 和 Langpop 作为信息来源。简单介绍一下, TIOBE 的排名是根据搜索引擎的点击来计算的。



全世界有多少 Java 开发者？ Ⅱ

Langpop 使用了更多的参数，除了搜索引擎的点击以外，它还使用了招聘信息，书籍标题，开源软件等等。把这些信息综合一下，我们可以得出，TIOBE 的数据显示 17% 的程序员使用 Java，Langpop 的数据显示 15% 的程序员使用 Java。取平均数 16%。4300 万 x16% = 688 万。

找工作门户

工作门户上面职位的数量和简历的数量是一个不错的信息来源。大型的工作门户同样提供劳动力市场报告，我们等会儿分析。

根据 Monster.com 的分析，大约 18% 的申请者会使用 Java。根据 Indeed.com 的信息，16% 的职位是跟 Java 相关。取平均数 17%。4300 万 x17% = 731 万。

软件下载

每一个 Java 开发者都会使用工具。我们假设他们至少使用一种 JVM 和编译器。如果你知道哪个 Java 开发者不使用他们的话，请你立即让我知道。大部分人至少还会使用 IDE，应用服务器等等。所以我们可以看看这些工具的下载数量去估算 Java 程序员的数量。

我们通过 Zeroturnaround 2012 年春天的统计数据来获得不同软件的市场占有率。

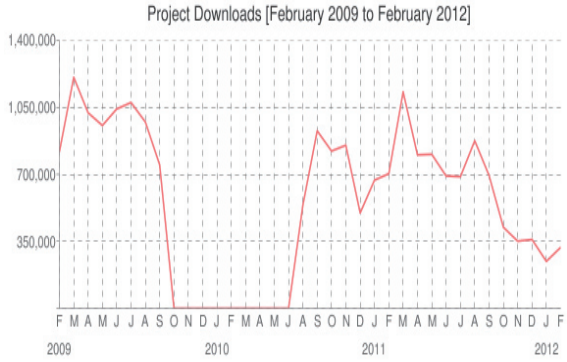
首先来看 Eclipse 的下载量。Eclipse Juno 是 6 月 27 日发布的，在头 20 天，已经被下载了 120 万次。根据 eclipse.org 发布的历史数据，Juno 将总共被下载 800 万次。过去的四次 Eclipse 主要发行版都是在 6 月份发布的，数据如下：

Juno – 800 万次下载（预估一年的数据，目前是头 20 天 120 万下载量）

Indigo – 600 万次下载；Helios – 410 万次下载；Galileo – 220 万次下载

取 Juno 和 Indigo 的平均值，我们可以估算 Eclipse 一年的下载量为 700 万次。根据 Zeroturnaround 的数据，Eclipse 占据 68% 的市场份额，那么我们可以估算大约有 1030 万的 Java 开发者。

下面来看看 Apache Tomcat 的下载情况。



根据相同的市场份额来源，有 59% 的 Java 开发者使用 Tomcat。那么我们在这里得到的数据是一共有 1118 万的 Java 开发者。

平均一下 Eclipse 和 Tomcat 的数据，我们可以得到 1074 万 Java 开发者。

总结

我们通过“工作市场，语言流行度，软件下载量”三种不同的数据来源进行估算，我们得到的数据有一些差距，688 万，731 万，1074 万。我们给这三个数字取平均，我们得到的数据时 831 万。虽然没有 Oracle 和 Wikipedia 声明的那么多，但是也接近，足够支撑一门生意，为 Java 开发者提供开发工具。

开发人员眼中最好的代码编辑器

如果我们把不同的程序开发人员比作三国演义中的各路诸侯大将的话，那么代码编辑器绝对可以称之我们手中的神兵利器，不同类型的开发人员使用的”兵器“也大有不同。好比兵器来说，没有绝对强的，也没有绝对好的，每一中兵器都有不同的优点和缺点，虽说俗话说的好，一寸长，一寸强，不过如果你没事去那都提着 " 关老爷 “的 “青龙偃月刀” 得瑟，貌似也不是很方便。那么对于我们这些开发人员来说，究竟什么样的代码编辑器是最好的呢？

在今天的文章中，我们将从以下几个方面来比较各种类型的代码编辑器，评判指标包括：

友好度；功能性；扩展性；界面 / 体验；跨平台；价格

大师级别

vi



vi 对于使用过 unix 的朋友来说，绝对是再熟悉不过的代码编辑器，有多少伟大的程序和代码是由 vi 开发编辑的啊，既然是大师级别的代码编辑器，对于我们这些普通人来说，只能说是好听不好用，基本上我周围的朋友使用 vi 的人大都是因为操作系统没有安装其它编辑器，也懒的花时间去安装。当然使用也相当麻烦了，你得记住一大堆的命令，如果你记不住，唯一能做的就是关闭。我现在还记得我初次使用 vi 的时候，自己老老实实的打印了一张命令表，贴在墙上随时参考使用。

Vim

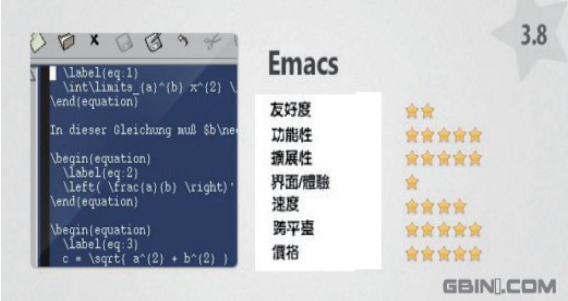
Vim 是一个类似于 Vi 的文本编辑器，不过在 Vi 的基础上增加了很多新的特性，Vim 普遍被推崇为类 Vi 编辑器中最好用的一个。最早 1991 年发布，赢得了开源世界的欢迎。和其它的代码编辑器相比不同的是命令行的工作方式。和简单的输入代码不同，你选择输入和选择文字，运行正则表达式的搜索，并且使用更多其它的命令。vim 使用脚本和插件可以变得非常适合扩展。可以支持 GUI 或者命令行。同时可以支持所有的操作系统。在大多数的 Linux 系统都预先装。



Emacs

Emacs 开发自 1970，现在依然开发。这个编辑器拥有扩展，并且可以加载自定义的类库。它是第一个实现了代码高亮，自动缩进和多编程语言支持的代码编辑器。和 Vim 一样，跨平台同时支持图形化界面和命令行。这个编辑器和 LISP 解析器整合，通过这个方式高手们可以修改它到极致。同时它是免费软件并且开源。

开发人眼中最好的代码编辑器 II



专业级别

Eclipse

Eclipse 是开发 java 应用的必备代码编辑器。这个 IDE 整合了插件结构，可以使得它轻松的支持其他编程语言。它拥有 C/C++，Ruby，PHP 和其它语言开发。类似 Google 的功能开发自己版本的开发套件，所以可以很简单的创建 Android 和 App 引擎。免费并且开源。



Apatana Studio



Aptana 是一个专门为富客户端 web 应用开发设计的代码编辑器。基于 Eclipse，并且帮定了强大的新工具。支持最流行的 web 开发语言：PHP，javascript，HTML，css，Ruby，Python 和其它更多插件。它拥有 Git 整合，能够部署你得应用到远程服务器。和 Eclipse 一样，Aptana 是免费和开源。

Netbeans

Netbean 是另外一个开发欢迎，和 Eclipse 一样，可以扩展支持其它的编程语言，PHP，Python，C/C++ 和其它。可以运行在 Linux，windows 和 OSX 上。Netbeans 可以快速的帮助你开发桌面应用，并且支持拖拽 GUI，带来的负面影响就是性能差一些。但是这个 IDE 免费并且开源



Dreamweaver



Dreamweaver 属于 adobe 应用套件之一，主要用来开发 web 应用。提供了最流行的 web 编程语言的支持：PHP，ASP.Net，Javascript，HTML，CSS。 主要为了初学者方便的编程，支持所见即所得的编辑方式。可以方便的部署到服务器，并且可以用来开发 jQuery 移动应用。同时支持 OSX 和 Window。单一价格 $399。当然买套件更加合算。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201207/350594.htm

做有中国特色的程序员

从出版业说起

网络作品排到靠前的，都不会太难看，一般人不爱看某部作品也是因为不喜欢这个类型，而此人也不会全不喜欢这些网络作品。究其原因，是因为网络作品都是让人先白看的，看的好了才出了头。而纸质作品就不一定了，排行榜靠前的，有好作品，也有垃圾。

博客园里有许多大牛都是写了博客，后来出了书。这些书也都不次，可能有人让为不好，是因为技术书不像小说，小说在读故事，技术书是在学知识或温习知识，有些技术书读得可能很违背某人的学习习惯，就感觉不好了。

所以说网络是块神奇的空间，很能造就高手，而网络高手是绝对不能小瞧的。

建筑工程

建筑工程不乏豆腐杂，但比起软件工程来说，豆腐杂的概率还是很小的。做软件，很可能某程序员在开发阶段一天没干一点正事，但在建筑行业，除非是收尾阶段，否则绝对不可能一个工人一天没干一点活儿。建筑工人还是很较挣钱的，但绝对是血汗钱。程序员还真不一定是血汗钱，因为软件工程没有建筑业成熟，常常因为分配不当，很多程序员每天在干无用功。建筑工人的门槛相对低一些，但也并非容易达到熟练水平，就算那个最低等的搬砖推灰的小工，至少想达到这个水准，你得身体力气跟得上，一月没有一天休息你得熬得住，想想都不容易。

中国软件的用处

软件不一定验收了就是有用的，在我看来，一款软件要想达到有用，至少实现以下功能之一：

1，为使用者节省了时间

2，为使用者节省了人力

3，为使用者带来了方便

4，为使用者带来了知识

5，使使用者能做到某种难以实现的事情

6，为使用者带来了快乐（比如游戏）

7，为使用者节省了资源

8，辅助使用者做出了判断

9，为使用者解决了某些问题

10，为使用者提供了资源

而有很多软件做出来以后，使用者不用吧已花了钱了，用吧什么也没带来，还得为学习这款软件而费时费力，得不偿失。

但有一个很悲观的消息，除了网络软件之外（包括网站），凡有用的软件中国人都习惯白去使用。而花了一大笔钱的软件，往往带来的好处不多。不是一点没有，是性价比太差。所以现状是：有用的软件没市场，有市场的软件用处有限。多尴尬的事。

而以上的一切带来的现状是：除了网络性软件外，中国的软件挣钱不容易。

程序员的被动

如果程序员做的东西要不挣不了钱，要不挣了钱反而让客户感觉不值，这样的话，程序员的价值就降低了。这样，建筑工人灌一天的水泥往往没白干，而程序员开发一周的工作往往白干

做有中国特色的程序员 II

这是常事儿。

这使程序的技术定位很尴尬，在中国，说句实话，不是每个程序员都能天天用到内存管理，就连最基础的算法也都被高级语言封装好了，当然也有很多低层的程序员，所以大家不用跟我争这个，最主要的是，高级语言、业务需求、工期决定了有很多编码与操作系统、算法、内存等等越来越脱节，这样的话，很多程序员干了若干年，特定业务熟悉了很多，但编程最基础的知识反而渐渐忘去。

这就出现了，让市面上很多程序员去做个基本的数据结构与算法（比如有向图，再比如好像是老赵说的排序），反而不会。这找谁说理去？还不如建筑工人呢，人家说会刷涂料真还做的不错。

有些公司，销售、行政、甚至实施人员都比程序员有优势，他们眼中这群天天如高僧般坐在那儿的人，占公司大半人数，反而做的贡献不多，反而让他们辛苦开拓的客户群都在谩骂。程序员，好被动。

不按套路出牌的中国人

中国到处都有不按套路出牌的现象：不用正版，不守时，造假，山寨，不排队，地沟油，不按合同办事，以次充好，没服务。这是种坏氛围，以至于每天每个人都习惯了这样规律，反而麻木了，不认为这是不对的，就如同我（我也不敢说你也是）每天白看人家的书白用人家的软件，而没有感觉自己有多少罪恶感。

中国程序员的出路

中国程序员也有做好的，虽然其最优秀的那些拿到国际上什么也不是，但毕竟能达到不被动，能用自己的努力使生活美好起来的还是很多的。总结一下，有如下几种：

1，教育型，主要是学知识，写书，讲课。

2，自由型，主要是为自己打工，做好的小团队软件，以积累用户而达到做有用的软件，挣有价值的钱。

3，业务人际型，一般都走向偏管理的路子，用高级语言，对数据库、报表、服务器和某个业务领域精通。

4，高级白领型，以进入国际大公司做核心项目为主，因为待遇好，属于打工仔里的高富帅。

5，创业型，当老板，或许这个已不是纯粹的程序员了，所以这个只能算半个。

以上几种路都可以走，但大多数程序员都还不在这 5 类之中，都是在某些公司不温不火的为生存努力着。

读技术书籍未必就真是无敌之出路

我常爱说，会中国常用的三千字写不了个好文章的话，学会中国八万汉字也未必就能写出来。同样，市面上的程序员如果去实现个初期的 Facebook 大多数还是能做到的，但你会这些知识还不能做出一个像 Facebook 一样有用的软件，那么你再精读了操作系统、设计模式、代码大全、算法导论 也还是照样做不出来。

所以我很反感网上列的程序员必读书目，一列就是五十大本，你花十年啃完这五十本，你还能写代码吗？

（本文未完，请阅读原文）■

http://developer.51cto.com/art/201207/350466.htm